J. Appl. Comput. Mech., 7(3) (2021) 1803-1813
DOI: 10.22055/JACM.2021.37664.3060

ISSN: 2383-4536
jacm.scu.ac.ir

Shahid Chamran University of Ahvaz

Journal of Applied and Computational Mechanics

Research Paper

Performance measure and tool for benchmarking metaheuristic optimization algorithms

François Schott[1], Dominique Chamoret[2], Thomas Baron[3], Sébastien Salmon[4], Yann Meyer[5]

[1]Percipio Robotics, Maison des Microtechniques 18, rue Alain Savary, Besançon, France, Email: francois_schott@orange.fr

[2] ICB UMR 6303, CNRS, UBFC, UTBM, Belfort, France, Email: dominique.chamoret@utbm.fr

[3]FEMTO-ST institute, Univ. Bourgogne Franche-Comté, CNRS, ENSMM Time and frequency dept., Besançon, France, Email: thomas.baron@femto-st.fr

[4]My-OCCS, Besançon, France, Email: Sebastien.Salmon@ar-e.com

[5]Univ. Savoie Mont Blanc, SYMME, FR-74000 Annecy, France, Email: yann.meyer@univ-smb.fr

Received June 08 2021; Revised July 02 2021; Accepted for publication July 05 2021.
Corresponding author: Dominique Chamoret (dominique.chamoret@utbm.fr)

Abstract. In the last decade, many new algorithms have been proposed to solve optimization problems. Most of them are meta-heuristic algorithms. The issue of accurate performance measure of algorithms is still under discussion in the scientific community. Therefore, a new scoring strategy via a new benchmark is proposed. The idea of this new tool is to determine a score, a measure of efficiency taking into account both the end value of the optimization and the convergence speed. This measure is based on an aggregate of statistical results of different optimization problems. These problems are judiciously chosen to cover as broad a spectrum of resolution configurations as possible. They are defined by combinations of several parameters: dimensions, objective functions and evaluation limit on dimension ratios. Aggregation methods are chosen and set in order to make the problem weight relevant according to the computed score. This scoring strategy is compared to the CEC one thanks to the results of different algorithms: PSO, CMAES, Genetic Algorithm, Cuttlefish and simulated annealing.

Keywords: Optimization algorithm, Performance Measure, Benchmark.

1. Introduction

An optimization problem consists of finding a set of valued decision variables which gives the best solution for a given problem. From a mathematical point of view, an unconstrained optimization problem can be formulated as a D-dimensional minimization problem (Equation (1)).

$$\min_{\mathbf{X}\in\mathcal{S}} F(\mathbf{X}) \quad \textbf{with} \quad \mathbf{X} = (\mathbf{x_1}, \cdots, \mathbf{x_D}) \tag{1}$$

In Equation (1), D, the dimension of the problem, is equal to the number of design variables. F is the objective-function to minimize and $\mathcal{S}$ is the space of possible values for $\mathbf{X}$, the variables. $\mathcal{S}$ is commonly known as the search space. Some constraints can be added but they are not considered in this work context. In the last few years, lots of new algorithms have been developed to solve the problem defined by Equation (1). Most of them are meta-heuristic ones [1, 2, 3].

From a theoretical point of view, it is accepted that no meta-heuristic can be considered as better than another one: this is the no free lunch theorem [4]. However, in practice, important differences in performance can be observed depending on the quality of the algorithm's mechanisms and the structure of the problem.

When a new algorithm is developed, most of the time, it is compared to a few others based on the means and standard deviations of the final results of a few test functions [5]. Furthermore, the selected test functions are often part of well-known function test suites, such as the one proposed by De Jong [6]. This a priori knowledge may lead to an erroneous assessment of the algorithm's efficiency. To avoid erroneous conclusions, algorithms should be tested on benchmarks [5, 7]. Since 2005, some reference benchmarks have been proposed in the literature [5]. Two of them are widely used: the CEC [8] and the Black-Box Optimization Benchmarking (BBOB)[9]. More recently, a platform designed to compare continuous optimizers in a black box has been developed: Comparing Continuous Optimizers (COCO) [10]. More details on this approach can be found in [11, 12]. CEC is composed of functions with higher dimensionality than BBOB but without noise [13]. The landscapes of CEC functions are very different from each other.

CEC is a fixed-budget scenario: the problem solving 'time' is fixed (It consists of stopping the algorithm after a certain number of evaluations of the objective function). Usually, it is expressed in function evaluation. It can also be expressed in CPU time or in clock time [14]. This approach might not be relevant for a real problem. Indeed, calls can be high CPU-time consuming and should therefore be reduced to a minimum. BBOB is a fixed-target scenario [9]. This approach raises an important issue: how to deal with an algorithm that does not reach the target or in too long a time ? Moreover, a fixed-target scenario (Strategy which consists in stopping the algorithm when a value of the objective function is reached) can be an inappropriate choice when associated with a meta-heuristic [15]. Indeed they can modify the number of function calls during the search.

The proposed benchmark works according to the same pattern as CEC or BBOB. An algorithm is run on a set of test functions from which data are extracted to provide a performance measure. The performance measure, which is the main score, is a general efficiency measure [16]. It may be exploited by researchers who wish to compare, tune or improve their algorithms. This score emphasizes the ability to fairly evaluate algorithms with criteria matching industry issues. This is made possible by the techniques used for computation. To summarize the raw results into a simple indicator, aggregation techniques [17] are used. To do this, the data must be comparable and it is therefore necessary to convert them into a same metric. This metric is a level of desirability [18, 19] obtained by using the Harrington desirability function [20]. Using a global level of satisfaction based on different metrics has already been done by [21] to optimize the tuning parameters of evolutionary algorithms. One of the originalities of this article is to introduce this concept to evaluate the main score of an algorithm. On this basis, this article defines a new benchmarking technology evaluating the performance of optimization algorithms and applies it to a representative set of optimization processes. The objective is to provide an efficiency measure in order to compare and rank algorithms. The idea is to propose a generic tool that allows standard comparisons in a real industrial context. The following sections of this article are organized as follows: An overview of the control parameters of our benchmark is made in Section 2.. Section 3. presents the performance measure and the tool for benchmarking. Finally, the scores of the selected algorithms are reviewed in Section 4..

2. Development of the proposed benchmark

2.1 Problem statement

The principle of benchmarks is quite simple: run optimization algorithms on reference examples, retrieve and analyze the results by computing a score. Two entities are therefore necessary to build a benchmark:

1. A set of test cases. In the majority of cases, the algorithms to be studied are tested on a set of well-known mathematical functions called "test functions". Note that for most of these functions, the global optimum is known. This makes it possible to simply compute the number of evaluations (i.e. the number of calls to the objective functions). This also allows to discuss the best value obtained for the objective function, thus allowing the introduction of a stopping criterion.

2. A score. After running the various tests on the reference functions, a score is calculated.

In an industrial context, a benchmark has to be able to evaluate the optimization algorithms in order to select the most efficient algorithm. An efficient algorithm must find a good objective function value in a short time. Therefore, a suitable compromise between those two wills has to be found [3]. Actually, the main score of an industrial benchmark can be considered as a measure of return on investment. An algorighm will be efficent if it is able to find a global optimum quickly, without loosing time in the determination of local optimum.

2.2 Control parameters

Before evaluating an optimization algorithm, a set of reference data must be defined. The main idea is to run the algorithm tested with a set of parameters fixed on optimization cases in order to measure its performance. An optimization case will be entirely defined by three control parameters:

1. an objective function, F.

2. a dimension, D.

3. a coefficient, *MaxFEs*, to regulate the maximum number of evaluations of the objective function (*FEs*) as given in Equation (2).

$$FEs = 10000 \times D \times MaxFEs \tag{2}$$

In this equation, 10000 is a reference value issued from classsical benchmark especially CEC [22] as the possible values of *MaxFEs*.

2.2.1 Objective function

The benchmark functions are used by the algorithm as black boxes and are explicitly known by the scientific community. In order to test the algorithm efficiency in a large context, various categories of functions must be considered to represent a wide range of difficulties. The choice of functions is of great importance in the design of a benchmark [16]. As advised by [16], a standard test set will be chosen, even if it is not perfect [13]. In this paper, our focus is on the 15 functions selected according to their characteristics in the CEC 2015 competition [23]. The details and expressions of this set of functions can be found in Appendix A.

2.2.2 Dimension

For example, in the building design industry, most optimization problems have between 8 and 24 variables, with an average of 15 [2]. In other fields the average number of variables may be higher. Still, considering our context, it seems wise to choose dimensions inferior to 50. Four different dimensions of the search space are tested for all functions: 10, 20, 30 and 50.

2.2.3 Maximum number of evaluations

Seven different values of *MaxFEs* are defined in the benchmark: 0.01, 0.02, 0.05, 0.1, 0.2, 0.5 and 1.

2.3 Run and draw

Due to the usual stochastic nature of the tested algorithms, several independent runs are needed. So, it is obvious that the significance of the results should be tested with an appropriate statistical measure. In this way, several evaluations are conducted on the same optimization case. It has been proved that $N_T = 51$ evaluations are enough to make relevant performance differences with a statistical significance [24]. This set of runs of the same optimization case is called a draw.

2.4 Stopping criteria

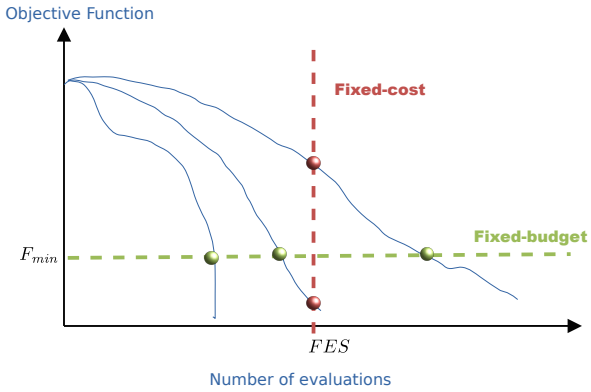

Fig. 1. Stopping conditions

The optimisation process for a case i ends when a stopping criterion is reached. Before talking about the result, the following problematic must be mentioned: which are the conditions that make the resolution of this case i stop and will allow to recover the data required to set up our indicators ? Considering the most classical convergence graphs which plot the evolution of the ojective function (o be minimized) against the number of this function evaluations, two ways to measure the performance can be used [25].

- The "Fixed target" strategy consists in stopping the algorithm when a value of the objective function is reached.
- The "fixed budget" strategy consists of stopping the algorithm after a certain number of evaluations of the objective function, noted FES

As explained by [26], both of these approaches, while of merit and justified by practical needs, have limits making them inappropriate for real-world if used individually. For instance, designers are not always able to define a target or a budget that makes sense. Some of these approaches' drawbacks might be solved. For instance, in fixed-target scenario, if the target value can not be reached, a success rate can be calculated as in [27]. Still as not all drawbacks cannot be tackled at one time, an interesting approach is to combine both scenario as suggested by [14]. In this case, target and costs are no longer defined as goals but as limits, which is an easier task to achieve for the designer. This is why, in an industrial context, the best method is to use several stopping criteria. [2] presents several stopping criteria and mentions the possibility of using several of them together. Based on this, three stopping criteria have been chosen for the benchmark proposed (Figure 1):

- The maximum number of evaluations of the objective function (*FEs*) defined in Equation (2) is reached.
- The target (F_{min}) is reached.
- One of the algorithm's potential stopping criterion is reached.

Finally, a run is stopped if at least one of these three criteria is reached. This stopping multi-criteria is another originality of the developed benchmark.

2.5 Global architecture of the proposed benchmark

To clarify our approach, the global architecture of the proposed benchmark is given in Figure 2. All the elements of the proposed benchmark are listed in Table 1. Finally, this benchmark is composed of $N_C = 420$ optimization cases. For statistical purposes, $N_R = 21420$ problems are performed to establish the score of the tested optimization algorithm.

Table 1. Control parameters of the benchmark

Element	Number	Value
Objective functions	$N_F = 15$	$\begin{cases} N_{uni} & = 2 \\ N_{multi} & = 7 \\ N_{hybrid} & = 3 \\ N_{comp} & = 3 \end{cases}$
Dimension	$N_D = 4$	$D \in \{10, 20, 30, 50\}$
$MaxFEs$	$N_M = 7$	$MaxFEs \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\}$
Optimizations cases	$N_C = 420$	$N_C = N_D \times N_M \times N_F$

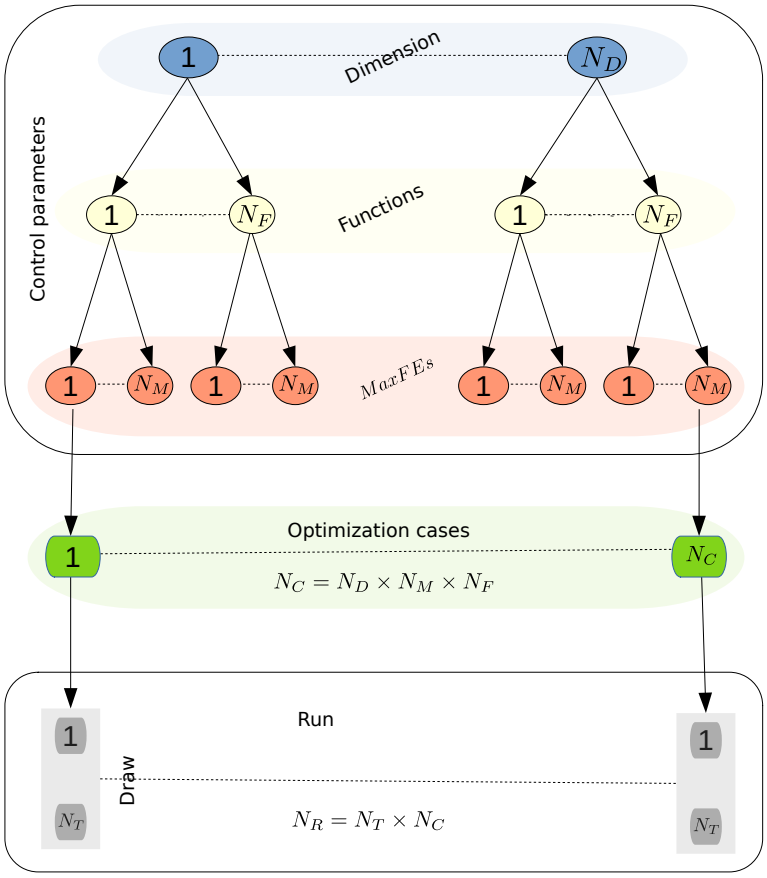

Fig. 2. Global architecture of the proposed benchmark

3. Performance measure and tool for benchmarking

As previously specified, the objective of the proposed benchmark is to provide a score that constitutes a representative overview of the performance of an optimization algorithm. The influences of all dimensions, functions and *MaxFEs* are taken into account when evaluating this score.

3.1 Result of a run

The result of a run represents the performance of an optimization algorithm on a particular run. Obviously, this result should be based on the value of the objective function at the end of the optimization (V_G). However, it should also consider the number of calls made by the algorithm (E_G). Indeed, the aim of this metric is to mimic a return on investment by measuring the quality of the value obtained for the invested solving time. Figure 3 explains how V_G and E_G are used to make the result of a run. The final expression, R_R, is given by Equation (3).

$$R_R = \boldsymbol{\mathcal{A}_R}\left(\boldsymbol{\Upsilon}_E(E_G);\, \boldsymbol{\Upsilon}_V(V_G)\right) \tag{3}$$

In this expression, several mathematical operators are introduced: the aggregation operator, $\boldsymbol{\mathcal{A}_R}$, and the normalization operators, ⊠, which are explained in the following subsections.

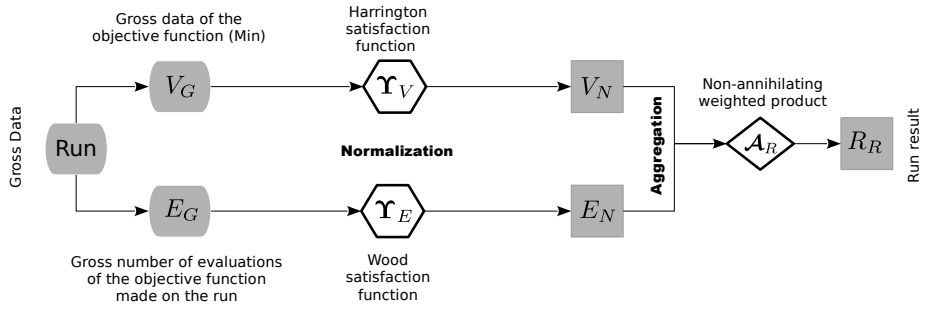

Fig. 3. Computation workflow of the result of a run

3.1.1 Normalization of the value of the objective function

The gross value of the objective function of an optimization case cannot be used as is in the determination of a performance indicator. In order to be able to implement a quantity that is as universal as possible, it is fundamental to be able to compare what is comparable. The gross value from the calculation run V_G is therefore normalized to obtain V_N.

To translate the performance of the algorithm, the idea is to introduce a level of desirability, as in [21]. When approaching zero, it reflects a value very far from the minimum of the function and has a higher risk not to lead to a correct solution. On the opposite, a level of desirability close to one reflects a minimum considered as acceptable. This aspect is accurately modeled by the Harrington function [20], called here Υ_V. This function is drawn on Figure 4. The Harrington function was selected in the case "the lower, the better" and a logarithmic scale was chosen, which corresponds to Equation (4).

$$\Upsilon_V(V_G)\begin{cases} V_N = exp(-exp(\eta+\beta\cdot\gamma)) \\ with\begin{cases} \beta = \dfrac{ln(ln(0.95)/ln(0.05))}{0.1-0.9} \\ \eta = ln(-ln(0.95)) - \beta\cdot 0.1 \\ \gamma = \dfrac{Log_{10}((V_G - F_{min})/F_{em})}{Log_{10}((F_{max}-F_{min})/F_{em})} \end{cases} \end{cases} \tag{4}$$

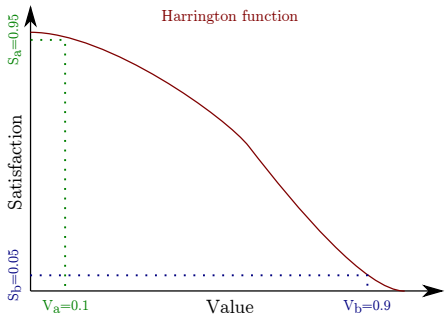

Fig. 4. Harrington Function

In Equation (4), F_{min} (resp F_{max}) is the minimum (resp. maximum) value of the objective function. In our case of the selected mathematical functions, F_{min} is well-known. F_{max} has been numerically estimated for each test function. Finally, the permitted error margin, F_{em}, is the threshold under which the minimum, F_{min}, is considered as reached. As in CEC 2014 [24], F_{em} has been set to $1e{-}8$. More details concerning this point can be found in [28].

3.1.2 Normalization of the number of evaluations

The variation 'The lower, the better' was selected because the lower the number of evaluations, the higher the satisfaction. A linear scale was chosen here as it works adequately where there are not many orders of magnitude between the minimum and maximum. Besides, using a steady method leads to a steadily satisfaction increase with the reduction of number of evaluations. Its normalization is presented in Equation (5).

$$\Upsilon_E(E_G) = E_N = 1 - \frac{E_G}{FEs} \tag{5}$$

3.2 Aggregation

An aggregation summarizes a set of data into a single indicator. When an optimization is performed in an industrial context, finding the best result in the shortest time is essential. An aggregation method with specific properties has been used: a non-annihilating version of the weighted product [17] has been chosen. The mathematical expression of this method is given in Equation (6).

$$\mathcal{A}_R : \begin{cases} R_R = 2\left(\left(\dfrac{1+V_N}{2}\right)^{w_1}\right) + \left(\left(\dfrac{1+E_N}{2}\right)^{w_2}\right) - 1 \\ and \quad w = \{0.75; 0.25\} \end{cases} \tag{6}$$

In Equation (6), w_1 and w_2 are used to weigh the quality of the value and the amount of time. The user can tune w according to the investigated optimization context. The more the gross value matters, the more the quality of value matters and the higher the w_1 value should be set. On the opposite, the higher the computer resources are, the less convergence speed matters and the lower the w_2 value should be set. Furthermore, if only the value quality (resp. convergence speed) is important, the value of w_1 (resp. w_2) should be set to 1. If $w_1 = 1$ (resp. $w_2 = 1$), only the value (resp. convergence) quality is taken into account for the computation of the result of a run. In this case, the result of a run is a measure of value (resp. convergence) quality, instead of efficiency measure. Furthermore, if the result of a run measures quality then the overall score also measures a quality instead of an efficiency. If the benchmark is used to guide the designer in the choice of an optimization algorithm, by comparing them in a competition, the w values have to be fixed. In this case, w should be tuned in a general context so that main score remains the general measure of efficiency as it is meant to be. The weights used in this paper have been chosen according to three main considerations: the weights should be balanced [3], the value should be weighted more than convergence [29] and the convergence speed is important [3].

3.3 Result of an optimization case

These results must take into account the stochastic nature of the tested algorithm. This is done by using reliability measures, by statistically evaluating the results of a random draw. Two reliability measures are introduced: the α result and the Ω result. The α score is the best result achieved in a draw. It represents the best possible outcome of an algorithm. On the opposite, the Ω score is the score achieved by a large majority of draw's runs. It represents what, at least, can be expected by the algorithm. For both scores, the scores of the runs are efficiency scores, meaning that they take into account the end value and the convergence speed.

3.3.1 α result

This result corresponds to the one that can be achieved if the optimization case can be solved many times. In this configuration, the best result, or the α result, will be kept. To this end, the best value of all runs is retained. The best sub-result, R_α, is defined as the maximum value obtained during all runs and is obtained with Equation (7)). In this expression, $\mathfrak{R}_R$ is the set of results of N_T runs of an optimization case.

$$\mathcal{A}_{\boldsymbol{\alpha}} \begin{cases} R_\alpha = \max\left\{\mathfrak{R}_R\right\} \\ \mathfrak{R}_R = (R_R^1, \cdots, R_R^{N_T}) \end{cases} \tag{7}$$

3.4 Ω result

In the situation of optimization and co-simulation, a run can be very CPU time-consuming. A last minute change in design specification puts a lot of strain on human resources [3]. In this situation, it is hardly possible to perform several runs to find a better solution than the previous one: only one run is done. The Ω result is the 95% confidence limit of the distribution of the draw results. The Ω sub-result, R_Ω, is defined in Equation (8). In this expression, $\mu_{\mathfrak{R}_R}$ (resp. $\sigma_{\mathfrak{R}_R}$) is the arithmetic mean (resp. the standard deviation) of the sample $\mathfrak{R}_R$.

$$\mathcal{A}_{\boldsymbol{\Omega}} \begin{cases} R_\Omega = \mu_{\mathfrak{R}_R} - y\ \sigma_{\mathfrak{R}_R} \\ \text{with } y \text{ such as } \left[\mu_{\mathfrak{R}_R} - y\ \sigma_{\mathfrak{R}_R}\ , +\infty\right] = 95\% \end{cases} \tag{8}$$

3.5 Result of an optimization case

The best and worst cases are aggregated by using the arithmetic mean of R_α and R_Ω as proposed in Equation (9). The proposed approach is summarized in Figure 6.

$$\mathcal{A}_{\boldsymbol{C}} \left\{ R_C = \mu(R_\alpha, R_\Omega) \right. \tag{9}$$

3.6 Global score computation

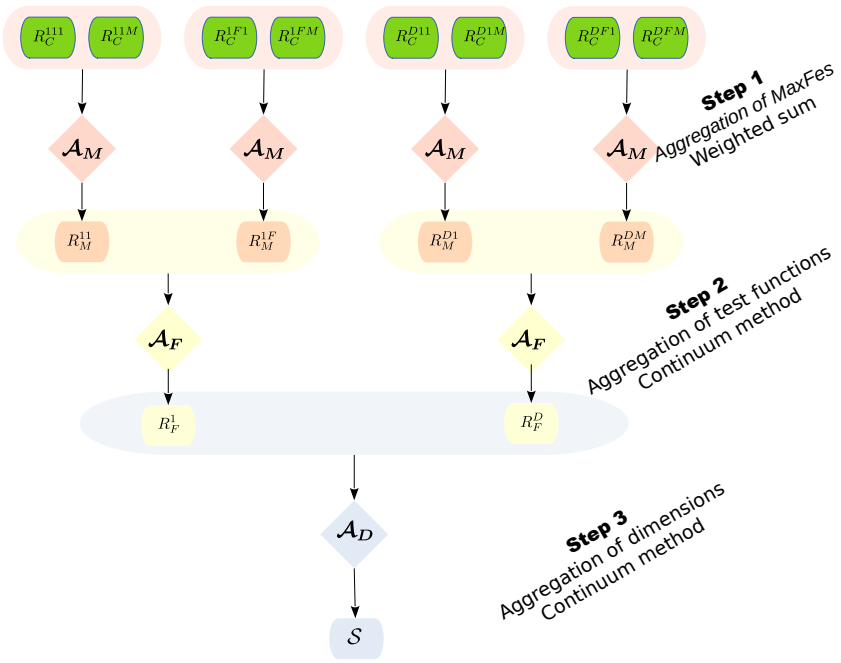

Fig. 5. Flowchart to compute a global score

In order to provide a global score, the results of the optimization cases must be summarized. The global score computation process is composed of three steps of aggregation as specified by Equation (10). A symbolic summary of this process is presented in Figure 5.

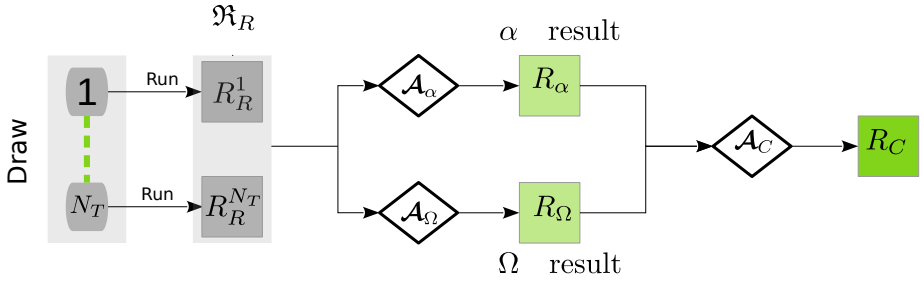

Fig. 6. Workflow to obtain the result of an optimization case

$$S = \boldsymbol{\mathcal{A}_D}\left(\boldsymbol{\mathcal{A}_F}\left(\boldsymbol{\mathcal{A}_M}(R_C)\right)\right) \tag{10}$$

- Step 1 - Aggregation of *MaxFEs* ($\boldsymbol{\mathcal{A}_M}$): All results of the optimization cases (R_C) are aggregated over *MaxFES*, giving $N_D \times N_F$ *MaxFEs* results (R_M);
- Step 2 - Aggregation of test functions ($\boldsymbol{\mathcal{A}_F}$): All *MaxFEs* results are aggregated over functions, giving N_D functions results (R_F);
- Step 3 - Aggregation of dimensions ($\boldsymbol{\mathcal{A}_D}$): To finish, all functions results are aggregated over dimensions, giving the main score (S).

3.6.1 Aggregation of *MaxFEs*

In an industrial context, if gains of the optimized solution are not sufficient to justify the investment of time, it is not worth computing. Thus, the higher the *MaxFEs* the higher the chance to have a useful result. Therefore, the higher the *MaxFEs*, the higher its weight in the aggregation method should be. The aggregation method retained is the weighted sum as given in Equation (11). In this equation, the superscript f refers to the fixed function (see Table 2) and the superscript d refers to the fixed dimension.

$$\boldsymbol{\mathcal{A}_M}: \begin{cases} R_M^{df} = \sum_{m=1}^{m=N_M} w_m \; R_C^{dfm} \\ \\ w_m = W_m / \sum_{m=1}^{m=N_M} W_m \\ with \quad W_m = \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1\} \end{cases} \tag{11}$$

In Equation 11, the W_m values have been chosen taking into account the recommendations from [26]. Piotrowski et al. mention that meta-heuristic optimization algorithms cannot successfully compete with mathematical programming methods when the computational budget is lower than a few hundred times the problem dimensions. From this remark, it can be deduced that *FEs* $= 100 \times D$ is the lower budget limit, i.e the limit number of evaluations of the objective function, for the relevant use of meta-heuristic optimization algorithms. Since the *MaxFEs* from 0.01 to 0.05 define this lower limit, they should have very low weights in the aggregation process. Piotrowski et al. [26] uses six computational budgets varying by two orders of magnitude. A similar choice was made here by using seven *MaxFEs* over two orders of magnitude. From this, it can be considered that the *MaxFEs* 0.5 and 1 are relevant choices for the use of meta-heuristic optimization algorithms, as confirmed by [30] which only uses those budgets to compute scores. Therefore, the *MaxFEs* 0.5 and 1 should be heavily weighted. The higher *MaxFEs* represent a little more than 50% and 25% of the final result and the lower ones only a few percent of this result.

3.6.2 Aggregation of test functions

The problems faced by companies are difficult, multi-modal and often close to hybrid and composite functions. Therefore, test functions should be weighted differently. Moreover, optimization is meaningless if the performance increase of the result is not high enough to justify the time and energy costs. As no algorithm can obtain convincing results on all problem topologies, an algorithm can be considered interesting if it can perform very well on some function typologies. Therefore, an algorithm with good or bad results should have a better score than another one with only average results. That is why a continuum method, with exponent superior to 1, has been chosen to aggregate over the test functions [17]. The mathematical expression of this method is given in Equation (12). The choice of weight values is based on experimental observations [28].

$$\mathcal{A}_F : \begin{cases} R_F^d = \sqrt{\sum_{f=1}^{f=N_F} w_f \left(R_M^{df}\right)^2} \\ w_f = W_f / (\sum_{f=1}^{f=N_F} W_f) \\ with \begin{cases} For \quad f = 1, 2; \quad W_f = 5/N_{uni} \\ For \quad f = 3, \cdots, 9; \quad W_f = 6/N_{multi} \\ For \quad f = 10, \cdots, 12; \quad W_f = 8/N_{hybrid} \\ For \quad f = 13, \cdots, 15; \quad W_f = 9/N_{composition} \end{cases} \end{cases} \tag{12}$$

3.6.3 Aggregation of dimensions

In an industrial context, it is more important for an algorithm to be efficient on high dimension problems than on low dimensional ones. The results from the high dimensional problems should be weighted more than the results from the low dimensional problems. In fact, an efficient algorithm for every dimension between 1 and 50 is searched. An algorithm with average results for each problem dimension should have a better score than an algorithm with both good and bad results. This is the reason why a continuum method with an exponent inferior to -1 has been chosen [17]. The mathematical expression of this method is given in Equation (13).

$$\mathcal{A}_D : \begin{cases} S = \sqrt[-2]{\sum_{d=1}^{d=N_D} w_d \left(R_F^d\right)^{-2}} \\ w_d = W_d / (\sum_{d=1}^{d=N_D} W_d) \\ with \quad W_d = \{10, 20, 30, 50\} \end{cases} \tag{13}$$

4. Scoring method analysis through algorithms testing

In order to analyze and compare the proposed benchmark to the CEC one, different algorithms were tested. For this task, a set of algorithms were chosen. The thorough selection of a consistent set of algorithms in order to test a benchmark is a work in itself. The motivation behind the choice of algorithms made is explained below. In order to fit the context of this work, the chosen algorithms should be classified as global search meta-heuristics. The algorithms represent different families of meta-heuristics as this benchmark is a generic tool. There is no guarantee that an up-to-date variant of a current optimization algorithm is a good representative of its family. Hence its initial version is used. Moreover, hybrid algorithms are not considered in this article because there are too many references in the literature. A limited number of algorithms have been selected to avoid burdening the reader with too much information. According to [31, 2, 5], particle swarm optimization, evolution strategy, genetic algorithm, swarm intelligence and differential evolution seem to be efficient families of optimization algorithms. Therefore, a representative from each of these families was chosen: the inertial PSO [15], the CMAES [32], an unpublished in-house Genetic Algorithm [33], the Cuttlefish [34] and DE/rand/1/bin [35]. A population-based version of the simulated annealing [36] was added to represent algorithms considered less efficient [2]. The PSO, GA and DE algorithms use a single stopping criterion, the radius which causes the algorithm to stop if, for a certain amount of iterations, all points evaluated are inside the selected radius. The CMAES, in addition of the radius stopping criterion, stops in case of a ill-conditioned matrix. The Cuttlefish and the SA algorithms do not use stopping criterion. Detailed information about the algorithms, such as their pseudo-code and their settings can be found in [28]. The proposed benchmark score is compared to the CEC one, considered as a reference and called here the classic benchmark. The algorithms results with both scoring methods are presented in Figure 7.

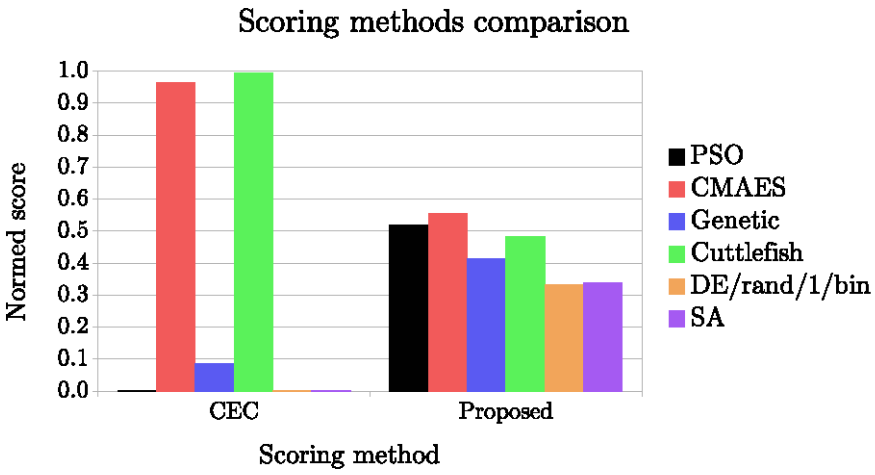

Fig. 7. Scores from the CEC benchmark and the proposed benchmark for classical meta-heuristic algorithms

From Figure 7, the CEC Benchmark shows very high scores for CMAES and Cuttlefish and very low scores for the other algorithms.

From the literature, this seems surprising. According to [2], evolutionary algorithms, such as CMAES, PSO and Cuttlefish should obtain good results and PSO should be the best one. In accordance with [5], PSO and GA are the most prolific families of algorithms in terms of publications since 2004. It suggests that the scientific community considers them to be efficient. In [31], Yang confirms this assertion and recognizes that PSO and GA have become the hotspots of current research in optimization and computational intelligence.

The proposed benchmark presents more homogeneous scores with limited maximum values. According to the results obtained, this benchmark is in good agreement with the literature. The CMAES algorithms show strong potential [32]. For both benchmarks, the assertion is confirmed and the CMAES is the reference algorithm. Due to their architecture and settings, GA, DE/rand/1/bin and SA are based on a large part of randomization. This explains why they have the lowest results. Cuttlefish and PSO have close results because they both use the displacement to the best local and global result. In fact, they are based on a well balance between random and logic. Finally, CMAES is the best optimization algorithm because it uses the most sophisticated technique to collect information about the iterated values of the objective function. An important source of improvement in optimization is the algorithm hybridization. To go further in this work, testing hybrid algorithms should be considered with regards to the results obtained for the initial algorithms. For instance, a PSO-CMAES hybridization, as proposed by [37], could obtain better results.

5. Conclusion

In the last few years, a great number of new optimization algorithms, most of them are meta-heuristic, has been developed. A hot topic in the community is to be able to measure the efficiency of these algorithms, to rank them and to select the right one for a given problem. As demonstrated by [5], benchmarking is the correct way to reach these objectives. Two benchmarks are quite renowned: CEC and BBOB. While of merit, their scoring strategy could be enhanced.

A benchmark, based on the CEC one, has been proposed. It is composed of 420 optimization cases with 51 runs per case. The cases are generated by the variation of three quantities: an objective function, a dimension and a *MaxFEs* coefficient. A scoring strategy has been developed to consider industrial needs and avoid some errors in the analysis of the algorithms. Several stopping criteria are used: target, *FEs* and algorithm convergence criteria. The results of a run take into account a normalized value for the objective function at the end of optimization and a normalized convergence speed. The results for the optimization cases are computed using statistics over the results of a run. The cases result goes through three steps of aggregation to produce a score.

Finally, the scoring method has been analyzed. First, several meta-heuristic algorithms have been tested on the CEC scoring method and the proposed scoring methods. These algorithms are PSO, CMAES, Genetic, Cuttlefish, DE/rand/1/bin, and SA. The idea is, using both benchmarks, to investigate if some conclusions, found in the literature, are confirmed by the algorithms scores. It appears that, with the proposed benchmark, the concluding comments are verified whereas with the CEC method they are not. Secondly, a thought experiment has been conducted to see the impact of using non-normalized results. It revealed that using the CEC scoring method is biased.

To properly use the proposed benchmark, one should keep in mind its limits. It has been designed for global search metaheuristic algorithms. The main score is a measure of efficiency and not a measure of quality [16]. This benchmark has been designed for industrial needs. Thus, its design is centered around the designer's point of view. A version of this benchmark including sub-scores could be found in [28]. These sub-scores could be used to analyze an algorithm and select an algorithm from a set of tested algorithms for a given problem.

Acknowledgments

This work was partly supported by the FEDER project SMART-INN under grant FC0001257 - 11002. The authors also like to thanks Camille Bourgin and Johann Novak for English proofreading.

References

[1] Boussand, I., Lepagnot, J., Siarry, P., A survey on optimization metaheuristics, *Information Sciences*, 2013, 237, 82 – 117, prediction, Control and Diagnosis using Advanced Neural Computations.
[2] Nguyen, A.T., Reiter, S., Rigo, P., A review on simulation-based optimization methods applied to building performance analysis, *Applied Energy*, 2014, 113(Supplement C), 1043 – 1058.
[3] Roy, R., Hinduja, S., Teti, R., Recent advances in engineering design optimisation: Challenges and future trends, *CIRP Annals*, 2008, 57(2), 697 – 715.
[4] Wolpert, D.H., Macready, W.G., No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation*, 1997, 1(1), 67–82.
[5] García-Martínez, C., Gutiérrez, P.D., Molina, D., Lozano, M., Herrera, F., Since cec 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness, *Soft Computing*, 2017, 21(19), 5573–5583.
[6] De Jong, K.A., Analysis of the behavior of a class of genetic adaptive systems, Ph.D. thesis, University of Michigan, USA, 1975.
[7] Liu, Q., Gehrlein, W.V., Wang, L., Yan, Y., Cao, Y., Chen, W., Li, Y., Paradoxes in numerical comparison of optimization algorithms, *IEEE Transactions on Evolutionary Computation*, 2020, 24(4), 777–791.
[8] Cheng, R., Li, M., Tian, Y., Zhang, X., Yang, S., Jin, Y., Yao, X., Benchmark functions for the cec 2017 competition on many-objective optimization, Tech. rep., University of Birmingham, UK, 2017.
[9] Hansen, N., Auger, A., Finck, S., Ros, R., Real-parameter black-box optimization benchmarking 2010: Experimental setup, Research Report RR-7215, INRIA, 2010.
[10] Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D., Coco: a platform for comparing continuous optimizers in a black-box setting, *Optimization Methods and Software*, 2021, 36(1), 114–144.
[11] Hansen, N., Auger, A., Mersmann, O., Tusar, T., Brockhoff, D., Coco: A platform for comparing continuous optimizers in a black-box setting, 2016, arXiv e-prints, arXiv:1603.08785.
[12] Hansen, N., Tusar, T., Mersmann, O., Auger, A., Brockhoff, D., Coco: The experimental procedure, 2016, arXiv e-prints, arXiv:1603.08776.
[13] Garden, R.W., Engelbrecht, A.P., Analysis and classification of optimisation benchmark functions and benchmark suites, *2014 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 1641–1649.
[14] Weise, T., Chiong, R., Lassig, J., Tang, K., Tsutsui, S., Chen, W., Michalewicz, Z., Yao, X., Benchmarking optimization algorithms: An open source framework for the traveling salesman problem, *IEEE Computational Intelligence Magazine*, 2014, 9(3), 40–52.
[15] Chamoret, D., Salmon, S., Di Cesare, N., Xu, Y.J., Bsg-starcraft radius improvements of particle swarm optimization algorithm: An application to ceramic matrix composites, P. Siarry, L. Idoumghar, J. Lepagnot, eds., *Swarm Intelligence Based Optimization*, Springer International Publishing, 166–174.
[16] Beiranvand, V., Hare, W., Lucet, Y., Best practices for comparing optimization algorithms, *Optimization and Engineering*, 2017, 18(4), 815–848.
[17] Collignan, A., Méthode d'optimisation et d'aide à la décision en conception mécanique : Application à une structure aéronautique, Ph.D. thesis, Université Sciences et Technologies - Bordeaux I, 2011.
[18] Gomes, W.J., Beck, A.T., Lopez, R.H., Miguel, L.F., A probabilistic metric for comparing metaheuristic optimization algorithms, *Structural Safety*, 2018, 70, 59–70.

Nomenclature

D	Dimension of an optimization problem (Number of variables)
F	Objective-function
X	Variables vector
x_i	i-th variable of X
$\mathcal{S}$	Search space
FEs	Number of objective function evaluations (stopping criteria)
MaxFEs	Maximum function evaluations
N_D	Number of dimensions
N_F	Number of test functions
N_M	Number of *MaxFEs*
N_C	Number of optimization cases
N_T	Number of runs of an optimization case
N_R	Total number of runs
N_{uni}	Number of uni-modal functions
N_{multi}	Number of multi-modal functions
N_{hybrid}	Number of hybrid functions
$N_{composite}$	Number of composite functions
F_{min}	Minimum of the objective function
F_{max}	Maximum of the objective function
F_{em}	Allowed error margin of the objective function
V_G	Gross value obtain by the algorithm at the end of a run
E_G	Gross number of evaluations made by the algorithm at the end of a run
V_N	Normalized value
E_N	Normalized number of evaluations
R_R	Run Result based on aggregate of V_N and E_N
R_C	Result of a case of optimization
R_α	α result
R_α	Ω result
$\mathfrak{R}_R$	Set of an optimization case's run results
S	Global score
$R_C^{df\,m}$	Case result for dimension d, function f and *MaxFEs* m
R_M^{df}	*MaxFEs* intermediate results for dimension d and function f
R_F^d	Function intermediate results for dimension d
$\boldsymbol{\Upsilon}_E$	Normalization operator for the gross number of evaluations
$\boldsymbol{\Upsilon}_V$	Normalization operator for the gross value
$\boldsymbol{\mathcal{A}_R}$	Aggregation operator used to compute run result
$\boldsymbol{\mathcal{A}_\alpha}$	Aggregation operator used to compute a case alpha sub-result
$\boldsymbol{\mathcal{A}_\Omega}$	Aggregation operator used to compute a case omega sub-result
$\boldsymbol{\mathcal{A}_C}$	Aggregation operator used to compute a case result
$\boldsymbol{\mathcal{A}_M}$	Aggregation operator over *MaxFEs*
$\boldsymbol{\mathcal{A}_F}$	Aggregation operator over functions
$\boldsymbol{\mathcal{A}_D}$	Aggregation operator over dimensions

[19] Huang, Y., Jin, Y., Ding, Y., New performance indicators for robust optimization over time, *2015 IEEE Congress on Evolutionary Computation (CEC)*, 1380–1387.

[20] Quirante, T., Sebastian, P., Ledoux, Y., A trade-off function to tackle robust design problems in engineering, *Journal of Engineering Design*, 2013, 24(1), 64–81.

[21] Mobin, M., Mousavi, S.M., Komaki, M., Tavana, M., A hybrid desirability function approach for tuning parameters in evolutionary optimization algorithms, *Measurement*, 2018, 114, 417 – 427.

[22] Qu, N.H.A.M.Z.A.P.N.S.J.J.L.B.Y., Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization, Tech. rep., Nanyang Technological University, Singapore And Jordan University of Science and Technology, Jordan And Zhengzhou University, Zhengzhou China, 2016.

[23] Qu, B., Liang, J., Wang, Z., Chen, Q., Suganthan, P., Novel benchmark functions for continuous multimodal optimization with comparative results, *Swarm and Evolutionary Computation*, 2016, 26, 23–34.

[24] Liang, J., Qu, B., Suganthan, P., Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2013, 635.

[25] Hansen, N., Auger, A., Brockhoff, D., Tusar, D., Tusar, T., COCO: performance assessment, *CoRR*, 2016, abs/1605.03560.

[26] Piotrowski, A.P., Napiorkowski, M.J., Napiorkowski, J.J., Rowinski, P.M., Swarm intelligence and evolutionary algorithms: Performance versus speed, *Information Sciences*, 2017, 384, 34–85.

[27] Boskovic, B., Brest, J., Protein folding optimization using differential evolution extended with local search and component reinitialization, *Information Sciences*, 2018, 454-455, 178 – 199.
[28] Schott, F., Contribution to robust and fiabilist optimization application to the design of acoustic radio-frequency filters, Ph.D. thesis, Université de Bourgogne Franche Comté, UTBM, 2019.
[29] Tawfik, A.S., Badr, A.A., Abdel-Rahman, I.F., Comparative optimizer rank and score: A modern approach for performance analysis of optimization techniques, *Expert Systems with Applications*, 2016, 45, 118–130.
[30] Chen, Q., Liu, B., Zhang, Q., Liang, J., Suganthan, P.N., Qu, B., Problem definition and evaluation criteria for cec 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization, Tech. rep., Congress on Evolutionary Computation (CEC), 2013.
[31] Yang, X.S., Koziel, S., Leifsson, L., Computational Optimization, Modelling and Simulation: Past, Present and Future, *Procedia Computer Science*, 2014, 29(Supplement C), 754 – 758.
[32] Hansen, N., The cma evolution strategy: a comparing review, *Towards a new evolutionary computation*, Springer, 2006, 75–102.
[33] Holland, J.H., *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, USA, 1992.
[34] Eesa, A., Mohsin Abdulazeez Brifcani, A., Orman, Z., A novel bio-inspired optimization algorithm, *International Journal of Scientific & Engineering Research*, 2013, 4, 1978–1986.
[35] Neri, F., Tirronen, V., Tirronen, v.: Recent advances in differential evolution: a survey and experimental analysis. artif. intell. rev. 33(1-2), 61-106, *Artif. Intell. Rev.*, 2010, 33, 61–106.
[36] Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., Optimization by simulated annealing, *Science*, 1983, 220(4598), 671–680.
[37] Muller, C.L., Baumgartner, B., Sbalzarini, I.F., Particle swarm cma evolution strategy for the optimization of multi-funnel landscapes, *2009 IEEE Congress on Evolutionary Computation*, 2685–2692.

Appendix A Objective-functions set

The set of objective-functions used by the proposed benchmark is presented by table 2. This set is the one used by the CEC 2015 [30]. The functions have been rotated, shifted and offset. The uni-modal and multi-modal functions use either one function or the sum of several functions. An hybrid function is obtained by the addition of the results of several sub-functions each using only a randomly selected part of all the variables. A composite function is obtained by the addition of the results of several sub-functions each given a weight depending on how close to the minimum of the function the point evaluated is. The exact formulation of the objective-functions composing this set could be found in [28]. The set of objective-functions used by the proposed benchmark is presented by table 2. This set is the one used by the CEC 2015 [30]. The functions have been rotated, shifted and offset. The uni-modal and multi-modal functions use either one function or the sum of several functions. An hybrid function is obtained by the addition of the results of several sub-functions each using only a randomly selected part of all the variables. A composite function is obtained by the addition of the results of several sub-functions each given a weight depending on how close to the minimum of the function the point evaluated is. The exact formulation of the objective-functions composing this set could be found in [28].

Table 2. Objective-functions set

Categories	No	Functions
Unimodal	1	Bent Cigar
	2	Discus
Multimodal	3	Weierstrass
	4	Schwefel
	5	Katsuura
	6	HappyCat
	7	HGBat
	8	Griewank plus Rosenbrock
	9	Scaffer
Hybrid	10	Hybrid 1 (Schwefel, Rastrigin, High Conditioned Elliptic)
	11	Hybrid 2 (Griewank, Weierstrass, Rosenbrock, Scaffer)
	12	Hybrid 3 (Katsuura, HappyCat, Griewank, Rosenbrock, Schwefel, Ackley)
Composite	13	Composition 1 (Rosenbrock, High Conditioned Elliptic, Bent Cigar)
	14	Composition 2 (Schwefel, Rastrigin, High Conditioned Elliptic)
	15	Composition 3 (HGBat, Rastrigin, Schwefel, Weierstrass, High Conditioned Elliptic)